Multilingual Theme Prediction for Customer Reviews Using Deep Pre-Trained Embeddings and Siamese Networks

Chinmay Prabhakar, Vindhya Singh, Wasiq Rumaney
PD Dr. Georg Groh, Gerhard Hagerer
Technische Universität Munich

July 30, 2019

1 Introduction

1.1 Motivation

Social networks have given a voice to nearly anyone and everyone with the increased spread of internet access. Over a course of the past few years, with multitude of opinions over the internet, there has been a growing interest in sentiment analysis and opinion mining in order to leverage these for analysis purposes. Simultaneously, many curated data-sets have been made publicly available for the same. Both these events have been driving factors for diving deep into the world of NLP opinion mining.

The work proposed here is done as a part of the NLP Lab Course Praktikum with the objective of finding the Multilingual Theme Prediction for Customer Reviews Using Deep Pre-Trained Embeddings and Siamese Networks. We have used the Amazon Product Review Data to train our model and have then used the Amazon Multi-Lingual Data for the testing purposes. The main objective achieved is, given product reviews, how well can our model predict the product category it talks about.

We have used Siamese Networks to find the similarity between the product reviews of the same product category. For transfer learning, we have used LASER Embeddings in our work to perform few shot learning on the German reviews of the test data set.In a nutshell, we have trained our model on the English product reviews and have tested it on product reviews written solely in German.

1.2 Data

1.2.1 Training Data

The data set which we have used in our work is the Amazon Product Review (8) for the purpose of training our model. Of the data-set provided, we have done theme prediction on 23 product categories. The overall data has about 10 Million product reviews. For our task we have used 230k of them. This number was decided based on the process of class balancing. Since the lowest number of reviews per product category available across all product categories was close to 10k (Musical Instruments), we took 10k samples per class and the resultant is the number 230k. The need for class balancing arose as the input data was too large in size and it lead to Google Colab failing on multiple occasions.

1.2.2 Test Data

The test data set we have used for this work is the Amazon Multilingual data for German language. The original data (7) has reviews written both in German and English. We have used **langdetect** (9) library to filter out the reviews written in any language other than German. Thus our test data has only reviews written in German.

One of the observations while working with **langdetect** was that there were many reviews in the test data that contained non-language words (terms that can not be identified by the langdetect library) such as only numerals or hyperlinks or simply blanks. It is strange as Amazon reviews are accepted only when they have been properly written. Our guess is that since it is a curated dataset over the decades, it may be the case that such reviews (dirty data) were written way back when the review requirements were not so stringent syntactically.

Again, due to limited powers of Google Colab, we have used only a subset of the test data. The total number of product reviews available in the Multi-lingual Amazon Product Review data is 615k. It is a highly unbalanced data-set with some product categories containing not more than 2 or 3 reviews. We have used 40k of the total number of reviews for the purpose of test.

It is worth noticing here that in our work we have used LASER embeddings (5). The technique that LASER utilizes involves working with bash scripts. We thus needed computer resources and were duly provided the same. However, in the last two weeks, towards the end of the Praktikum, the LRZ was down for maintenance. Thus, we limited our test data set to 40k only.

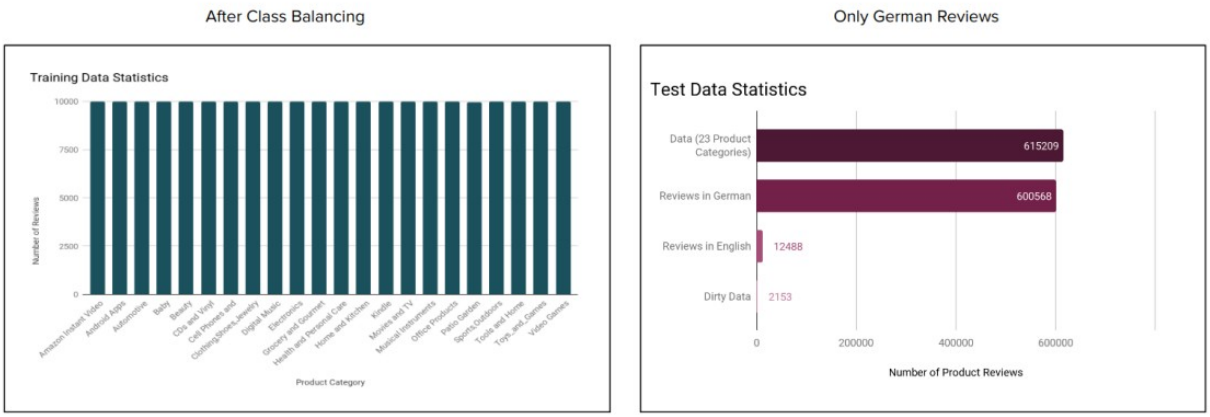

Figure 1: Data Statistics

2 Literature review

2.1 LASER

LASER (5) stands for Language-Agnostic SEntence Representations. LASER has been developed by Facebook and has now been open-sourced. It uses a Bi-LSTM Encoder-Decoder.LASER sets a new state of the art on zero-shot cross-lingual natural language inference accuracy for 13 of the 14 languages in the XNLI corpus.

LASER opens the door to performing zero-shot transfer of NLP models from one language, such as English, to scores of others — including languages where training data is extremely limited. LASER is the first such library to use one single model to handle this variety of languages, including low-resource languages, like Kabyle and Uighur, as well as dialects such as Wu Chinese.

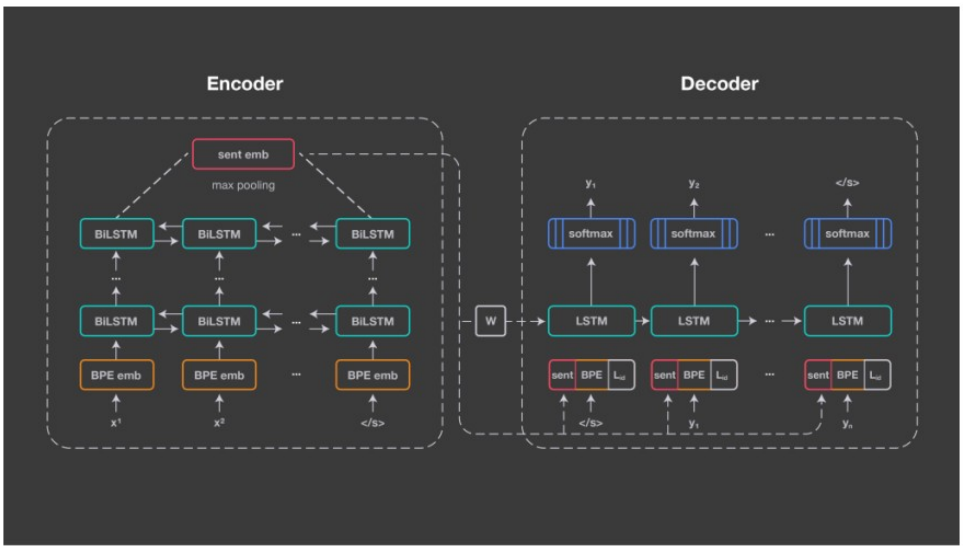

Figure 2: LASER Architecture

LASER also offers several additional benefits:

- It delivers extremely fast performance, processing up to 2,000 sentences per second on GPU
- The sentence encoder is implemented in PyTorch with minimal external dependencies.
- Languages with limited resources can benefit from joint training over many languages.
- The model supports the use of multiple languages in one sentence.
- Performance improves as new languages are added, as the system learns to recognize characteristics of language families.

They have used one shared encoder for all input languages and a shared decoder to generate the output language. The encoder is a five-layer bidirectional LSTM (long short-term memory) network. In contrast with neural machine translation, they do not use an attention mechanism but instead have a 1,024-dimension fixed-size vector to represent the input sentence. It is obtained by max-pooling over the last states of the Bi-LSTM. This enables the comparison of sentence representations and feeds them directly into a classifier.

2.2 Siamese Network

Siamese networks are a special type of neural network architecture. They build on the idea that the neural networks learns to differentiate between two inputs instead of the model learning to classify its inputs. Thus, it learns the similarity between the inputs.

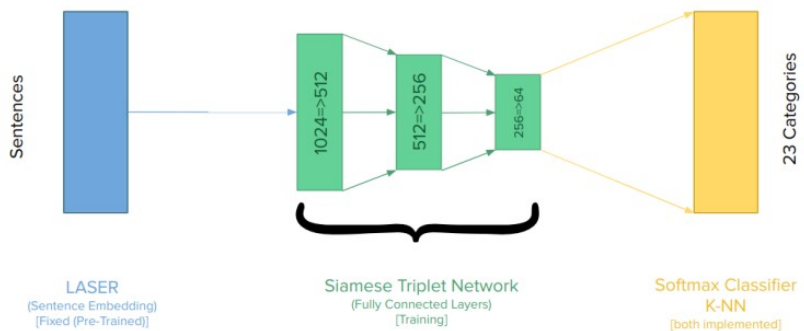

Figure 3: Architecture Used in this work

A Siamese network consists of two identical neural networks. Each of them takes one of the two input product reviews. The last layers of the two networks are then fed to a loss function (we use

triplet loss), which calculates the similarity between the two product reviews. It is to be noted here that the product reviews are represented as sentence level embeddings using LASER.

2.3 Triplet Loss and Online Triplet Mining

Online triplet mining was introduced in Facenet (3).The idea utilized is to compute useful triplets on the fly, for each batch of inputs. Given a batch of **B**, we compute the embeddings and we then can find a maximum of **B** triplets. Of course, most of these triplets are not valid (i.e. they don't have 2 positives and 1 negative).This technique gives more triplets for a single batch of inputs, and does not require any offline mining. It is therefore much more efficient. (6) Triplet loss makes sure that:

- Two examples with the same label have their embeddings close together in the embedding space
- Two examples with different labels have their embeddings far away.

Having said that, it will not be optimal to push the train embeddings of each label to collapse into very small clusters. The main aim is that given two positive examples of the same class and one negative example, the negative should be farther away than the positive by some margin.

The loss will be defined over triplets of embeddings: an anchor, a positive of the same class as the anchor, a negative of a different class For some distance on the embedding space *d*, the loss of triplet

$$\mathcal{L} = max(d(a,p) - d(a,n) + margin, 0) \tag{1}$$

We minimize this loss, which pushes *d(a,p)* to 0 and *d(a,n)* to be greater than *d(a,p)+margin*. As soon as *n* becomes an 'easy negative', the loss becomes zero.

For our case, we use semi-hard online triplet mining where semi-hard triplet is defined as:

$$d(a,p) < d(a,n) < d(a,p) + margin \tag{2}$$

3 Experiments

Expectations

Prepare data. Perform baseline experiments, i.e., the simplest possible way to solve the problem Download and use pre-trained models and source code. Make results consistent, comparable, and insightful.

Week 1 Results

Literature Survey, Data Exploration, Understanding the problem statement, finding related works.

Week 2 Results

In this week we tried to implement some already implemented models to understand the task practically and to get a gist of how to proceed with the task. We consulted some blogs and Github pages, the references of which have been duly acknowledged in the slides we presented. Going by the timeline of our expectations, we could successfully:

Pre-process the validation and testing data i.e Organic Data, Kaggle Dataset and Multilingual Dataset. Get the code running to generate the Word Embeddings on Word2Vec. Implement LSTM Siamese Network.

There were certain **Problems** that we faced:

The training data causes a problem for data loading purposes. Google Colab doesn't take a file as big as 32 GB. The Multi-lingual data is not precisely multi-lingual, with more records being in English rather than in German.

For the upcoming weeks, our task is to get the training data loaded and to start with our model training.

Week 3 Results

We used LASER (See Section 2.1) for generating the sentence embeddings. For our loss function, we are taking into account Triplet Loss. We have used semi-hard triplet mining in this work. (See Section 2.3)

We used 2.3k samples from the entire training data set to try and see if the model is working and if the data fits with our model. We tried over-fitting the model as is the customary first step to ensure that the network is performing as expected. We implemented dropout as well to ensure the training pipeline is up and running.

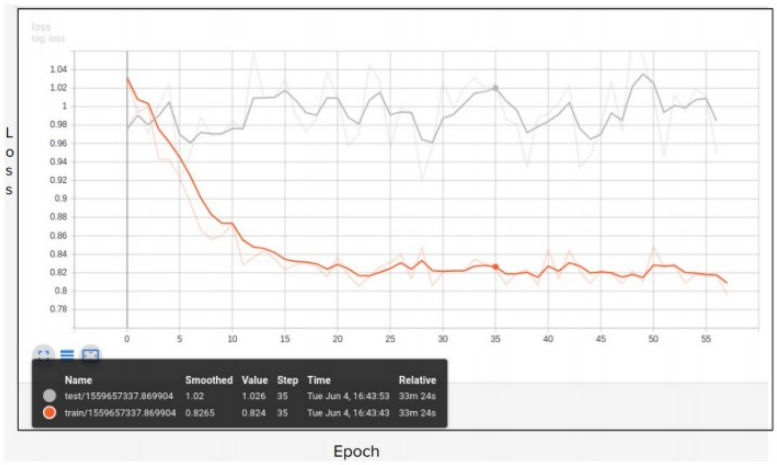

Figure 4: Dropout showing that our model is not given sufficient data yet but at least it is able to reduce the loss function.

Week 4 Results

Overfitting the model on a small data sample. Since we have 23 product categories, we took 100 samples from each and thus the resultant was 2.3k samples overall. The code runs perfectly and we have decent loss curves.Moving on from the sample dataset, our next task was to make sure that we start training on the entire corpus of data. However, the *problems* that arose were:

1) For the task of tokenization itself, our Colab kept running for >10 hours and failed

Solution: Use a Docker Container

2) However, the file sizes still creates a problem with processing time needed to get the sentence embeddings of the reviews.

Solution: Class Sampling. Since the Product Category with least reviews is Musical Instruments (10k reviews), we took 10k samples from each product category to get intermediate results.

Week 5 Results

This week we tried: hyperparameter tuning , Sentence Embeddings for the Test and Validation Data, Code Optimisation Implementing the 1-NN Algorithm over the topmost layer, Visualizing the sentence embeddings using t-SNE

Problems

The test data i.e. Multilingual Amazon Product Reviews data consists of dirty data and some English Reviews as well.We need to do the Zero shot transfer on the German Reviews Sentence Embeddings. Hence it is important for us to have only reviews written only in German.The test data consists of 615k reviews. We needed more computer power for the tasks of Tokenisation, Byte Pair Embeddings and for generating Sentence Embeddings. Finding Hyperparameters is, as always a cumbersome task.

Solutions

For using only reviews written in German, we used the langdetect library and as a result, the total number of German reviews came out to be 600k out of 615K of the total dataset.We used LRZ compute power (CPUs) that helped us in creating sentence embeddings for the test set. Usage stats in the appendix section. Ran and tested the F1 scores, Micro and Macro Averages to test the best hyper-parameters.

Results and Graphs in the next section.

4 Results

Our classifier works pretty well on the training data. The F1 scores per product category, the micro and macro average scores, as well as the accuracy reported are decent values as compared to the baselines.

	precision	recall	f1-score	support
Amazon Instant Videos	0.70	0.67	0.68	1664
Android Apps	0.70	0.71	0.71	1496
Automotive	0.49	0.45	0.47	1630
Baby	0.68	0.65	0.67	1570
Beauty	0.71	0.60	0.65	1606
CDs and Vinyl	0.57	0.60	0.59	1596
Cell Phones and Accessories	0.65	0.61	0.63	1579
Clothing, Shoes, Jewelry	0.68	0.63	0.65	1606
Digital Music	0.78	0.75	0.76	1615
Electronics	0.53	0.48	0.50	1604
Grocery and Gourmet	0.74	0.67	0.70	1639
Health and Personal Care	0.30	0.40	0.34	1537
Home and Kitchen	0.58	0.54	0.56	1612
Kindle	0.83	0.82	0.83	1537
Movies and TV	0.61	0.63	0.62	1574
Musical Instruments	0.63	0.61	0.62	1615
Office Products	0.73	0.87	0.79	3203
Patio Garden	0.64	0.58	0.61	1644
Pet Supplies	0.71	0.69	0.70	2076
Sports, Outdoors	0.33	0.42	0.37	1598
Tool and Home Improvement	0.44	0.43	0.43	1589
Toys_and_Games	0.73	0.64	0.68	1648
Video Games	0.82	0.83	0.82	1550
accuracy			0.63	38788
macro avg	0.63	0.62	0.63	38788
weighted avg	0.64	0.63	0.63	38788

Figure 5: F1 Scores per Product Category for the training data

Going forward, we wanted to ensure that our classifier works well. In order to visualise the results, we plotted the ROC Curve. As is visible from the plot, the ROC curves per product category is well above the diagonal, indicating good classification.

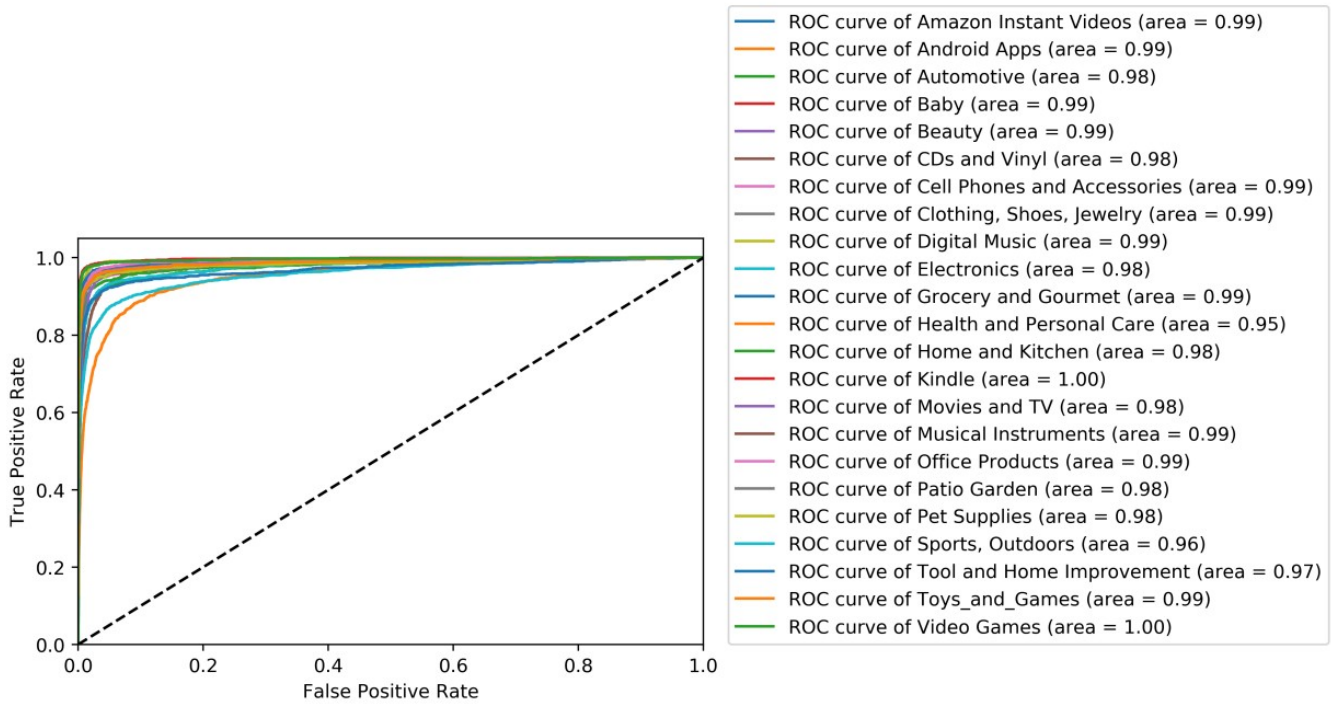

Figure 6: ROC Curve of the training data labels

One of the basic questions that arises is what were the values of hyper-parameters that we have used in the model? The reported best values are upon implementing Dropout as well.

Margin	Number of Dimensions	Accuracy (%)
7	64	7
3	32	4
3	64	63

Figure 7: Best values of the hyper-parameter

As a final result, we report here the comparison of the F1 scores for randomly picking samples in the test data set. We picked up samples in the variation of $n = 1, 10, 100$, per product category. As can be seen from the plot, the mean value of the accuracy values increases per step size. We can also observe that the accuracy significantly increases for the values of $n = 1$ to $n = 10$ but the rise in accuracy is not as significant from $n = 10$ to $n = 100$. Hence, one of the **pivotal results** of the experiment is that the best random sample size will be best taken at $n = 10$.

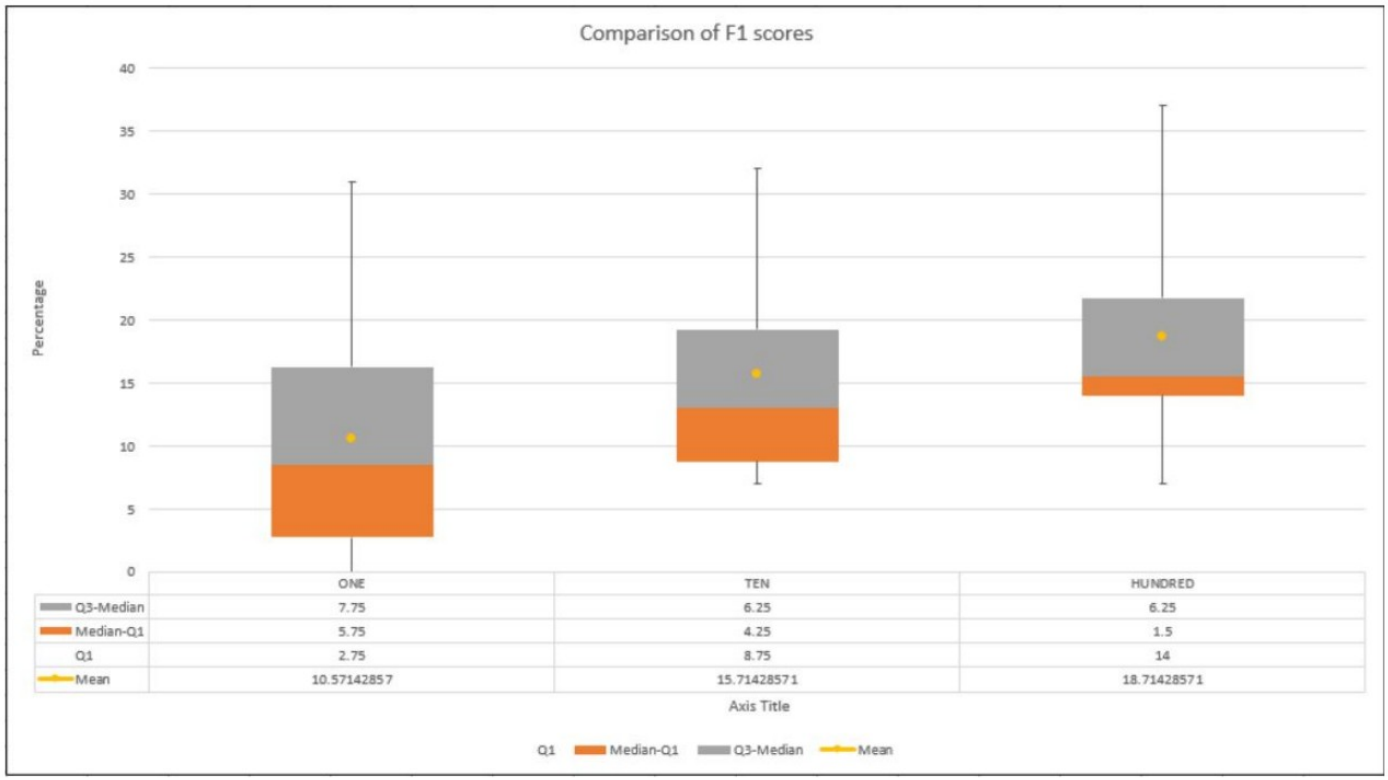

Figure 8: Box plot of comparison of F1 scores

For all the cases of the random sample size reported here, i.e. $n = 1,10,100$ we have also reported the Confusion Matrices and per product category F1 scores. [1]

	precision	recall	f1-score	support
Automotive	0.18	0.14	0.16	899
Baby	0.24	0.18	0.21	899
Digital_Ebook_Purchase	0.05	0.13	0.07	302
Digital_Music_Purchase	0.15	0.22	0.18	694
Digital_Video_Download	0.17	0.14	0.15	899
Electronics	0.24	0.30	0.27	899
Home	0.20	0.19	0.20	899
Mobile_Apps	0.17	0.13	0.15	899
Musical Instruments	0.38	0.36	0.37	899
Shoes	0.29	0.31	0.30	899
Sports	0.14	0.10	0.12	899
Toys	0.13	0.15	0.14	899
Video	0.16	0.12	0.14	899
Video DVD	0.15	0.18	0.16	899
accuracy			0.19	11784
macro avg	0.19	0.19	0.19	11784
weighted avg	0.20	0.19	0.19	11784

Figure 9: F1 scores per Product Category for the Test Data

[1] All confusion matrices, F1 scores per sample size are attached in the Appendix

We tried and implemented multiple things. Primarily, over the training data we wanted to ensure that the sentence embeddings are clustering well. For that purpose, we implemented the t-SNE Embeddings.

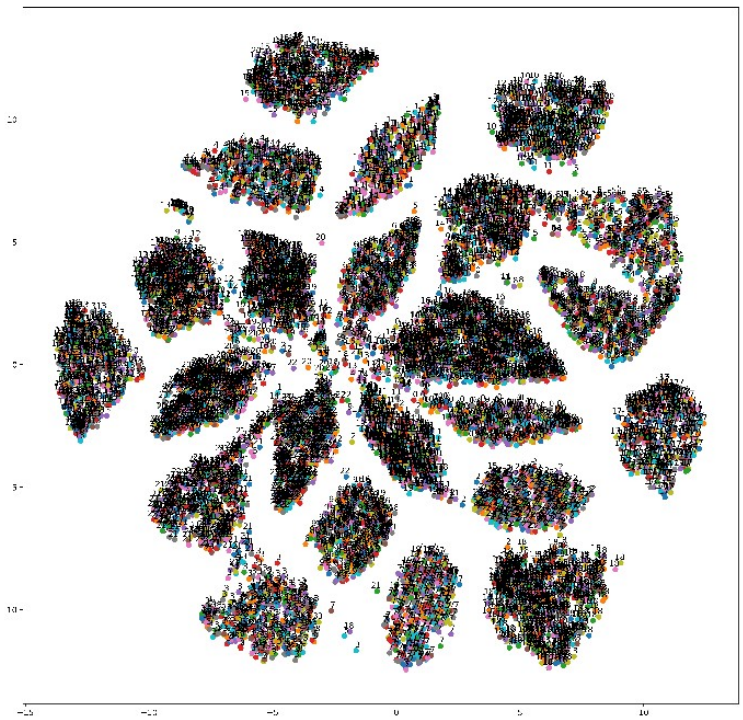

Figure 10: t-SNE Embeddings showing distinct clusters getting formed with labels

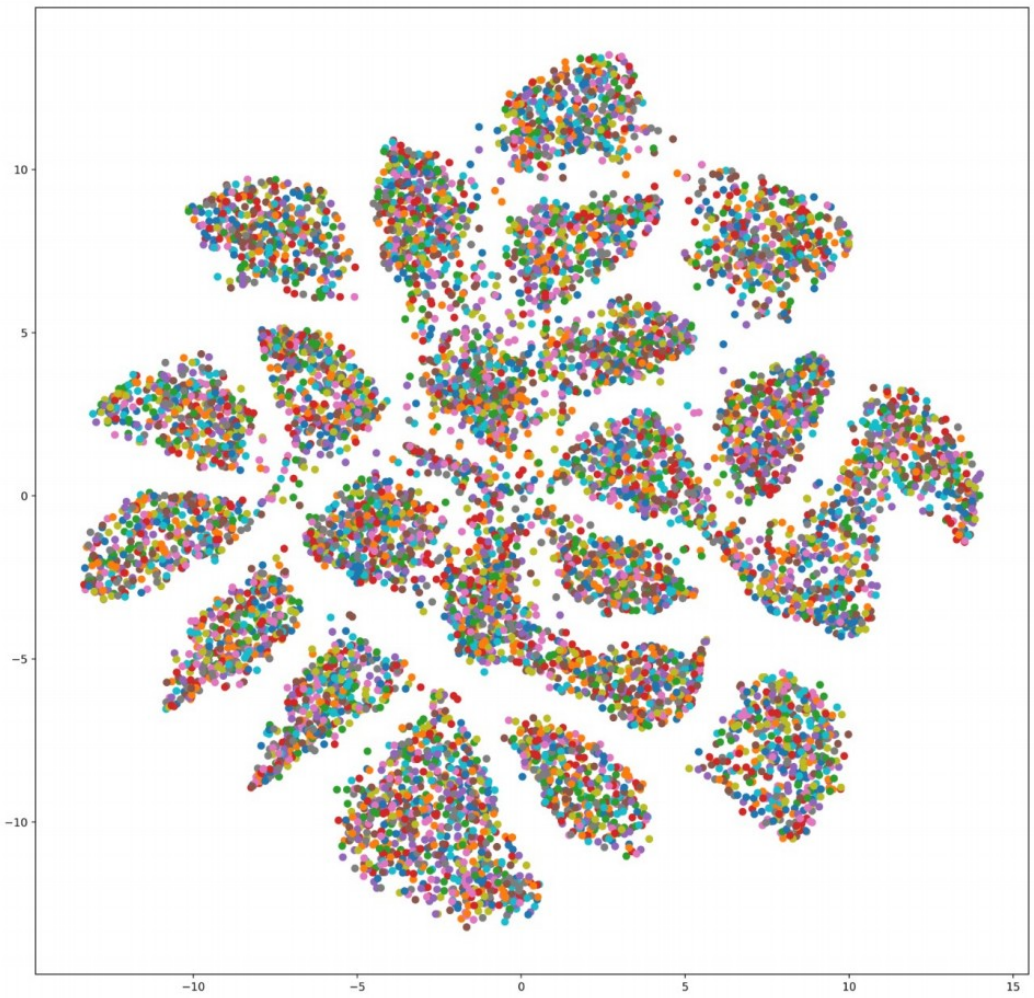

Figure 11: t-SNE Embeddings showing distinct clusters getting formed without labels

Please note that the colors in matplotlib are limited and with 23 product categories, not all clusters can be assigned unique colors. However, it can be clearly seen that the clusters are getting formed distinctly.

5 Analysis

We have reported accuracy on the Amazon German Review Dataset. However, the accuracy is around 20 percent only on 100 random samples collected, for k-nn (k = 5) and batch size = 20.
So we tried to deep dive on the possible reasons for this score.

- Check if the predictions made by the model are “sensible”
- Do some testing with dummy values

We tried using an English sentence and then translating the same sentence in German and checking it’s cosine similarity. The result of cosine similarity was 1.00 in this case. We tried working on sentences as whole and also on paragraphs. The paragraphs were also taken on a per sentence basis. The cosine similarity was 0.99994.

```
sample_sentence_en = "If you are reading this review it probably means you are a poor, unfortun
ate sole that was smart enough to get into to law school but not smart enough to get out. Now i
t is mere weeks before your Property final and you are freaking out about the Rule Against Perp
etuities. Just buy this. Do it. It's worth wasting another 50 bucks on yet another book. It rea
lly is. The examples are actually really helpful, unlike most casebooks."
sample_sentence_de = "Wenn Sie diese Rezension lesen, bedeutet dies wahrscheinlich, dass Sie ei
ne arme, unglückliche Sohle sind, die klug genug war, um an die juristische Fakultät zu kommen,
aber nicht klug genug, um herauszukommen. Jetzt ist es nur noch wenige Wochen, bis Ihr Eigentum
endgültig ist und Sie über die Regel gegen Ewigkeiten ausflippen. Kaufen Sie einfach das. TU e
s. Es lohnt sich, weitere 50 Dollar für ein weiteres Buch zu verschwenden. Ist es wirklich. Die
Beispiele sind im Gegensatz zu den meisten Casebooks wirklich hilfreich."
```

```
In [127]: cosine_sim(embed_triplet_1.detach().cpu().numpy().reshape(-1), embed_triplet_2.detach().cpu().n
          umpy().reshape(-1))

Out[127]: 1.0
```

Figure 12: Cosine Test for the English and German product reviews

So the vector embeddings appear together in the semantic vector space. Thus, we propose that LASER’s claims of being Language Agnostic hold true. However, we do need more data for our network to train upon. We could have reported the results on the increased dataset had LRZ not been down for maintenance.
We thus conclude that the network we use learns, and when introduced to one shot learning rather than zero shot, learns better and gives improved results.

6 Conclusion

To perform Theme Prediction, we propose a novel concept of using Siamese Networks with Triplet loss. We went a step ahead and used the model trained from English text and used it to make predictions on German text. The results were not as good as we expected but we believe that this can be improved with more training data. As future work, we intend to compare LASER with other embeddings such as XLING, etc.

PS: Code shared on GitLab-Group07

Git repository: https://gitlab.lrz.de/nlp-lab-course-ss2019/nlp-lab-group7

References

[1] Julian McAuley and Jure Leskovec. *From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise through Online Reviews*. CoRR, 2013.

[2] Julian McAuley and Jure Leskovec. *Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text*. RecSys, 2013.

[3] Florian Schroff and Dmitry Kalenichenko and James Philbin. *FaceNet: A Unified Embedding for Face Recognition and Clusterin*. CoRR, 2015.

[4] Predicting Amazon Product Review Helpfulness.
`https://bcourses.berkeley.edu/files/70257 249/download?download`$_frd = 1$

[5] Zero-shot transfer across 93 languages: Open-sourcing enhanced LASER library.
`https://code.fb.com/ai-research/laser-multilingual-sentence-embeddings/`

[6] Siamese and triplet learning with online pair/triplet mining.
`https://github.com/adambielski/siamese-triplet`

[7] Amazon Multi-lingual data on AWS S3.
`https://s3.amazonaws.com/amazon-reviews-pds/readme.html`

[8] Amazon Product Reviews
`http://jmcauley.ucsd.edu/data/amazon/`

[9] Langdetect library
`https://pypi.org/project/langdetect/`

Appendix

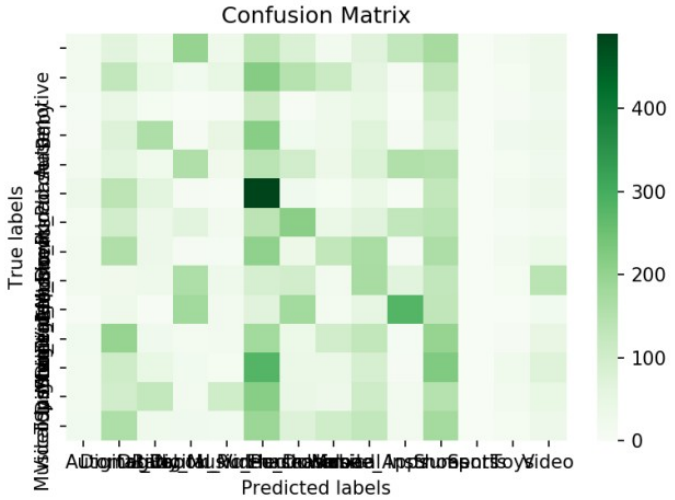

Figure 13: Confusion Matrix n=1, batchsize = 30

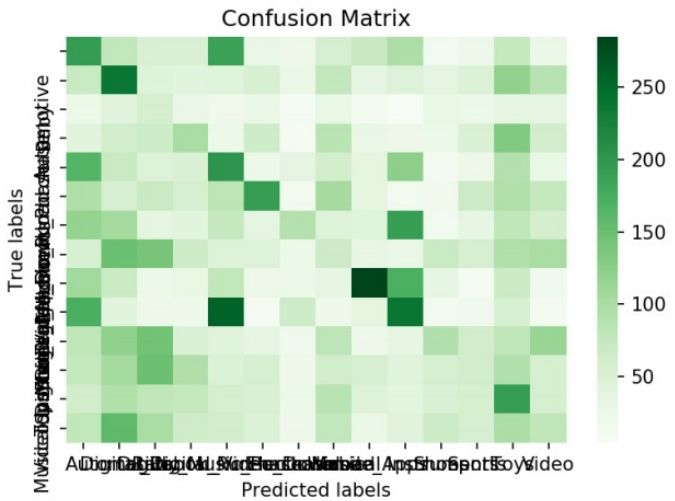

Figure 14: Confusion Matrix n=10, batchsize = 30

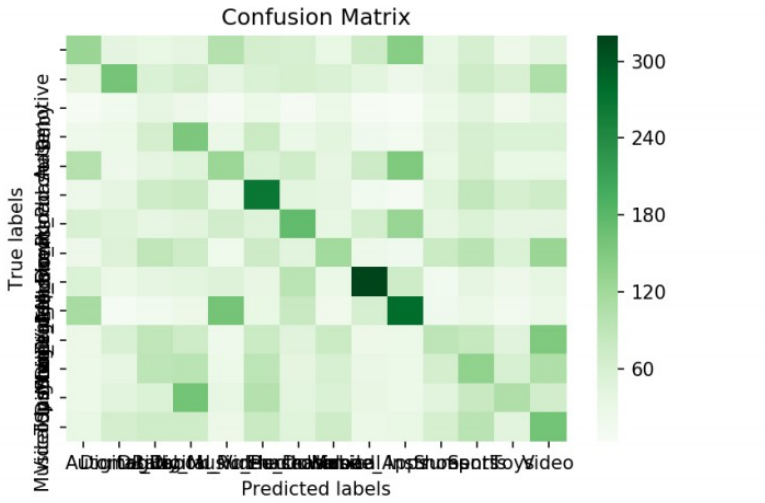

Figure 15: Confusion Matrix n=100, batchsize = 30

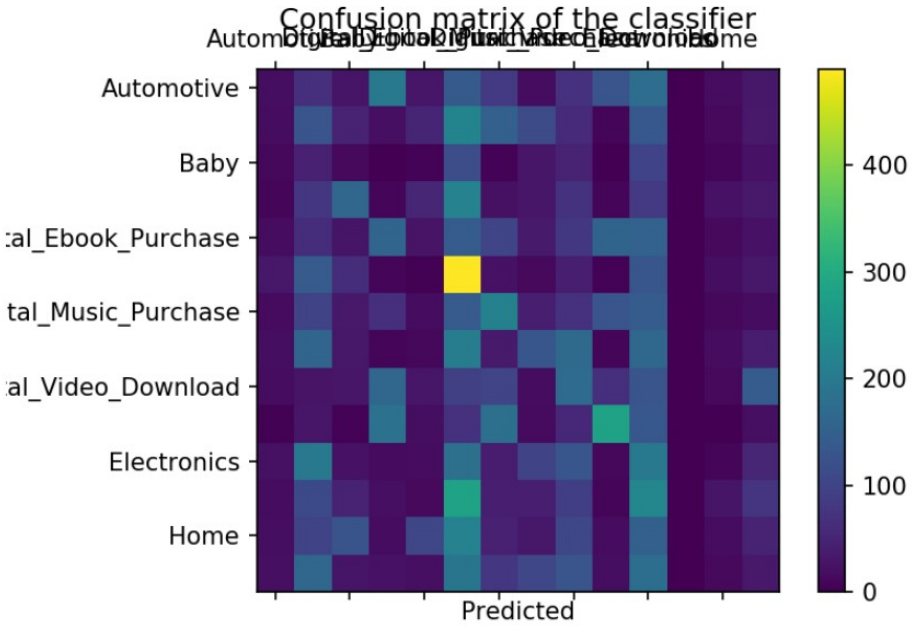

Figure 16: Classifier Confusion Matrix n=1, batchsize = 30

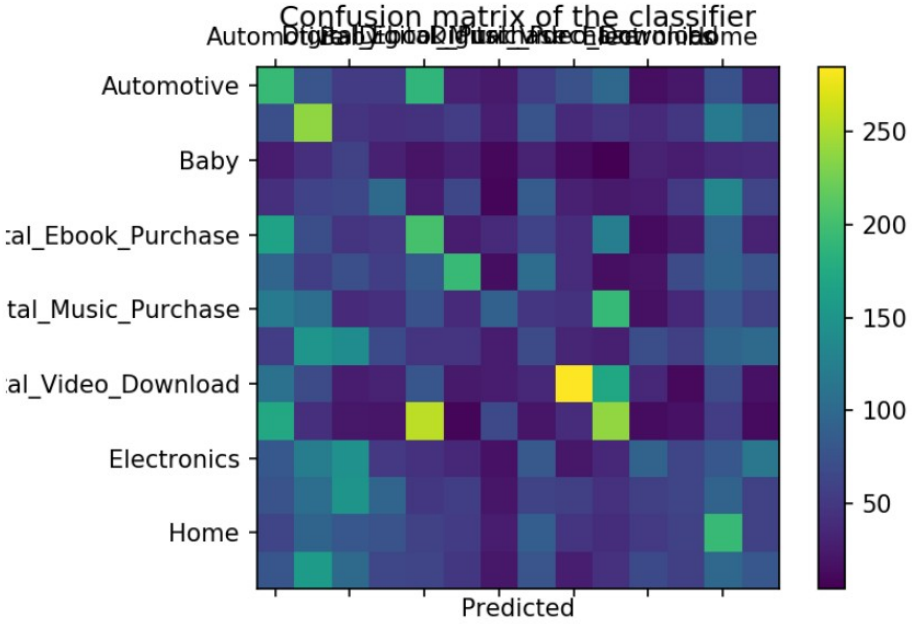

Figure 17: Classifier Confusion Matrix n=10, batchsize = 30

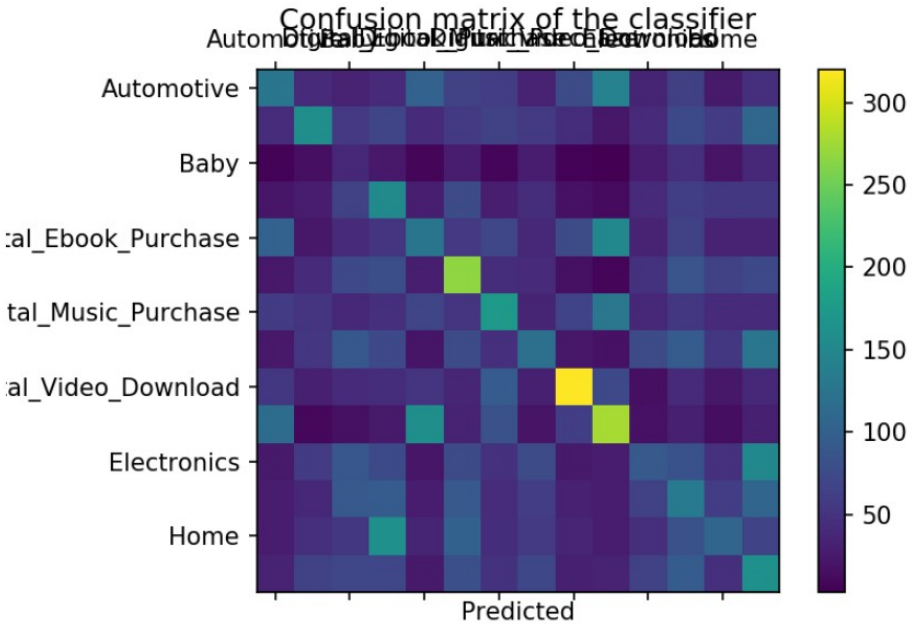

Figure 18: Classifier Confusion Matrix n=100, batchsize = 30

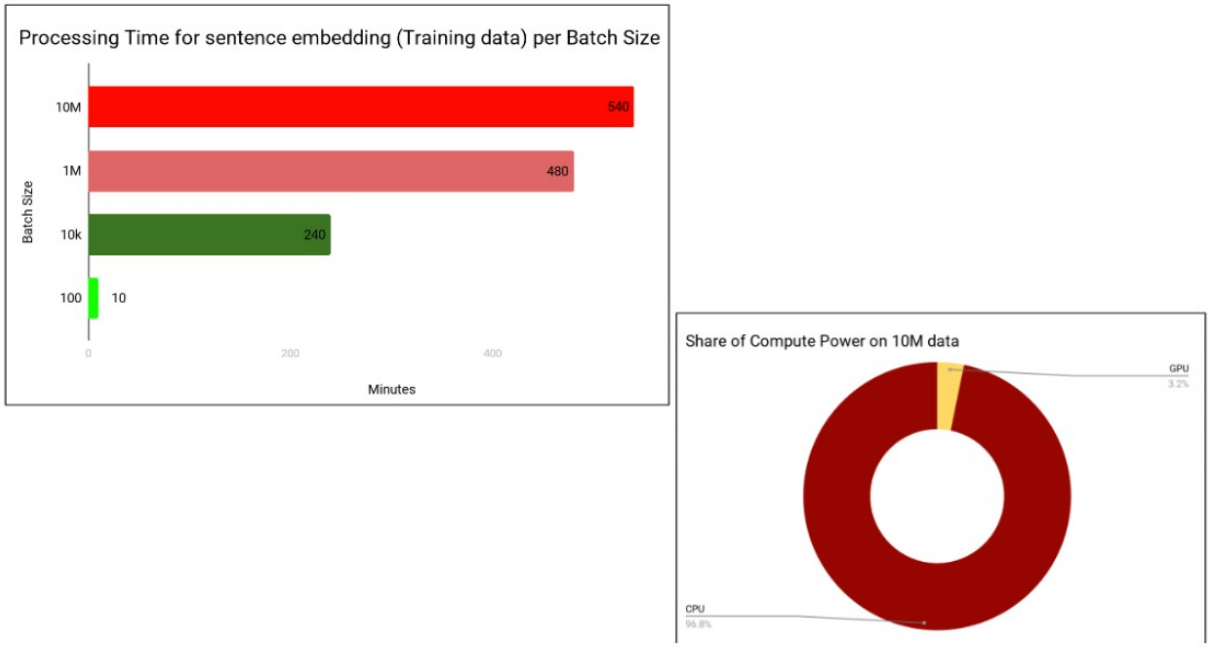

Figure 19: Process time stats

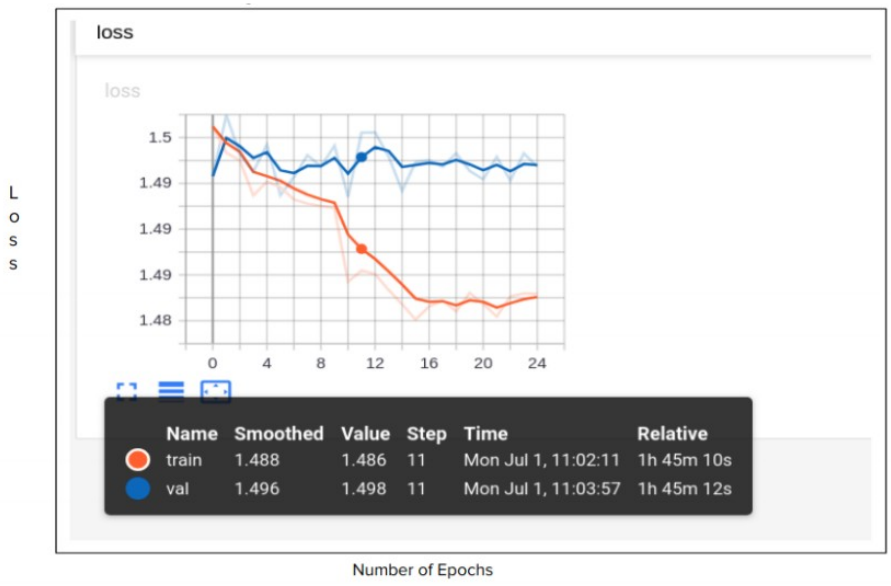

Figure 20: Final reported Loss Curve on the Test data with margin=0.3 and dropout

Statutory Declaration

Hiermit versichere ich, dass diese Arbeit von mir persönlich verfasst ist und dass ich keinerlei fremde Hilfe in Anspruch genommen habe. Ebenso versichere ich, dass diese Arbeit oder Teile daraus weder von mir selbst noch von anderen als Leistungsnachweise andernorts eingereicht wurden. Wörtliche oder sinngemäße Übernahmen aus anderen Schriften und Veröffentlichungen in gedruckter oder elektronischer Form sind gekennzeichnet. Sämtliche Sekundärliteratur und sonstige Quellen sind nachgewiesen und in der Bibliographie aufgeführt. Das Gleiche gilt für graphische Darstellungen und Bilder sowie für alle Internet-Quellen. Ich bin ferner damit einverstanden, dass meine Arbeit zum Zwecke eines Plagiatsabgleichs in elektronischer Form anonymisiert versendet und gespeichert werden kann. Mir ist bekannt, dass von der Korrektur der Arbeit abgesehen und die Prüfungsleistung mit nicht ausreichend bewertet werden kann, wenn die Erklärung nicht erteilt wird.

I hereby declare that the paper presented is my own work and that I have not called upon the help of a third party. In addition, I affirm that neither I nor anybody else has submitted this paper or parts of it to obtain credits elsewhere before. I have clearly marked and acknowledged all quotations or references that have been taken from the works of others. All secondary literature and other sources are marked and listed in the bibliography. The same applies to all charts, diagrams and illustrations as well as to all Internet resources. Moreover, I consent to my paper being electronically stored and sent anonymously in order to be checked for plagiarism. I am aware that the paper cannot be evaluated and may be graded “failed” (“nicht ausreichend”) if the declaration is not made.

Signature

München, 30.07.2019

Place, Date